МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |
|  |  |

**ОТЧЕТ ПО УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ**

(2023/2024 учебный год)

                                          Усов Александр Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики к.т.н., доцент, Карамышева Н.С.

*(должность, ученая степень, ученое звание)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |
|  |  |

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                            Усов Александр Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения                  1                 семестр                 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики к.т.н., доцент, Карамышева Н.С.

*(должность, ученая степень, ученое звание)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Планируемая форма работы во время практики | Количество часов | Календарные сроки проведения работы | Подпись  руководителя  практики от вуза |
| 1 | Выбор темы и разработка индивидуального плана проведения работ | 2 | 25.06.2024 -  25.06.2024 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 25.06.2024 –  27.06.24 |  |
| 3 | Разработка алгоритма | 45 | 27.06.24 –  1.07.24 |  |
| 4 | Описание алгоритма и программы | 16 | 30.06.24 –  1.07.24 |  |
| 5 | Тестирование | 6 | 01.07.24 –  01.07.24 |  |
| 6 | Получение и анализ результатов | 10 | 01.07.24 –  02.07.24 |  |
| 7 | Оформление отчёта | 14 | 01.07.24 –  04.07.2024 |  |
|  | **Общий объём часов** | 108 |  |  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЁТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                            Усов Александр Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

Усов А.С. выполнял практическое задание «Сортировка выбором». На первоначальном этапе были изучен и проанализирован алгоритм сортировки выбором, был выбран метод решения и язык программирования С, на котором была написана программа сортировки массива методом выбора. Также, осуществил работу с файлами. Протестировал и отладил программу. Оформил отчёт.

Бакалавр Усов А.С. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель Карамышева Н.С. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

                                            Усов Александр Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 25.06.24 по 08.07.24

Кафедра «Вычислительная техника»

В процессе выполнения практики Усов А.С. решал следующие задачи: тестирование алгоритма сортировки выбором, анализ работы алгоритма, сравнение существующих методов сортировки.

За период выполнения практики были освоены основные понятия и технологии сортировки выбором, реализован метод работы с файлами. Во время выполнения работы Усов А.С. показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Усов А.С. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики к.т.н., доцент, Карамышева Н.С. « » 2024 г.

# **Содержание**

Введение ……………………………………………………………………………………….. 7

1 Постановка задачи …………………………………………………………………….. 9

1.1 Достоинства алгоритма сортировки выбором …………………………………... 9

1.2 Недостатки алгоритма сортировки выбором …………………………………..... 9

2 Выбор решения ………………………………………………………………………. 11

2.1 Описание алгоритма …………………………………………………………..…. 11

3 Описание программы ………………………………………………………………... 13

4 Схемы программы …………………………………………………………………… 16

4.1 Блок-схема программы ………………………………………………………….. 16

4.2 Блок-схема алгоритма …………………………………………………………… 17

5 Тестирование программы …………………………………………………………… 18

6 Отладка ……………………………………………………………………………….. 20

7 Совместная разработка ……………………………………………………………… 21

Заключение …………………………………………………………………………………... 22

Список используемой литературы …………………………………………………………. 23

Приложение А. Листинг программы ………………………………………………………. 24

# **Введение**

Сортировка данных на сегодняшний день при современном развитии компьютерных технологий является одним из наиболее распространенных процессов современной обработки данных. Задачи на сортировку данных встречаются очень часто в различных профессиональных сферах деятельности.

Алгоритмы сортировки очень широко распространяются практически во всех задачах обработки информации. Они образуют отдельный класс алгоритмов, применяются с целью осуществления последующего более быстрого поиска.

Важность сортировки основана на том факте, что на ее примере можно показать многие основные фундаментальные приемы и методы построения алгоритмов. Сортировка является хорошим примером огромного разнообразия алгоритмов, которые выполняют одну и ту же задачу. Кроме того, многие из них имеют определенные преимущества друг перед другом. За счет усложнения алгоритма можно добиться существенного увеличения эффективности и быстродействия алгоритма по сравнению с более простыми методами. Как правило, термин сортировка понимают, как процесс перестановки объектов некоторого множества в определенном порядке.

Сортировка выбором (Selection Sort) — это простой алгоритм сортировки, который работает за время O(n²), делая его неэффективным для больших объемов данных. Он последовательно выбирает наименьший элемент из неотсортированной части массива и перемещает его в отсортированную часть. Основные недостатки включают высокую временную сложность и отсутствие адаптивности, что делает его медленным на больших наборах данных. Однако, сортировка выбором может быть полезна для небольших массивов или когда простота реализации важнее производительности, например, в образовательных целях для обучения основам алгоритмов сортировки.

# **1. Постановка задачи**

Поставленная задача: необходимо заполнить массив из n-ого количества элементов случайными числами, записать данные элементы в отдельный файл.

После этого выполнить сортировку вставками над данными, находящимися в массиве, записать отсортированные данные в другой файл, посчитать время выполнения и количество перестановок значений массива при сортировке.

Использовать сервис GitHub для совместной работы. Создать и выложить коммиты, характеризующие действия, выполненные каждым участником бригады.

Оформить отчет по проведенной практике.

## **1.1 Достоинства алгоритма сортировка выбором:**

* Простота и понятность реализации;
* Отсутствие дополнительных затрат памяти;
* Предсказуемое поведение;
* Минимальное количество обменов элементов;
* Эффективность для небольших массивов;
* Стабильность при выборе на основе определённых критериев;

## **1.2 Недостатки алгоритма сортировка выбором:**

* Низкая эффективность для больших массивов:
* Большое количество сравнений:
* Отсутствие адаптивности:
* Отсутствие стабильности:
* Отсутствие параллелизма:
* Неэффективность для больших наборов данных:
* Низкая кэш-эффективность:
* Сложность модернизации:

# **2. Выбор решения**

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. Си – это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью и переносимостью. Указанные преимущества Си обеспечивают хорошее качество разработки почти любого вида программного продукта.

В качестве среды программирования была выбрана программа Microsoft Visual Studio. Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

## **2.1 Описание алгоритма**

**Сортировка выбором**

Просто и незатейливо — проходим по массиву в поисках максимального элемента. Найденный максимум меняем местами с последним элементом. Теперь неотсортированная часть массива уменьшилась на один элемент, так как она больше не включает последний элемент, куда мы переставили найденный максимум. К этой уменьшенной неотсортированной части массива применяем те же действия: снова находим максимум и ставим его на последнее место в этой новой неотсортированной части. Повторяем процесс, пока неотсортированная часть массива не сократится до одного элемента. В результате массив будет полностью отсортирован, так как все максимальные элементы будут перемещены на свои правильные позиции в обратном порядке.

Идея сортировки:

1. В неотсортированном подмассиве ищется локальный максимум (минимум).
2. Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве.
3. Если в массиве остались неотсортированные подмассивы — смотри пункт 1.

# **3. Описание программы**

При запуске программы выводится меню из трех пунктов:

1. Сортировка определенного файла;

2. Сортировка файла с определенным количеством чисел;

0. Выход;

При выборе пункта 1. выводятся сообщения “Введите имя файла с несортированными числами (без расширения):” “ Введите имя файла с сортированными числами (без расширения):”. Пользователю необходимо ввести имя файла с заранее записанными не сортированными числами в командную строку, а затем ввести имя файла для записи туда сортированных чисел. Потом у пользователя спрашивают в каком порядке отсортировать числа: в прямом или обратном. Затем этого выводится время сортировки файла и кнопка возврата в главное меню.

При выборе пункта 2. выводятся сообщения “Введите количество чисел:” “Введите минимальное значение:” “Введите максимальное значение:”. После этого выводится “Введите имя файла с несортированными числами (без расширения):” “ Введите имя файла с сортированными числами (без расширения):”. Далее у пользователя спрашивают в каком порядке отсортировать числа: в прямом или обратном. Затем этого выводится время сортировки файла и кнопка возврата в главное меню.

Программа показывает меню, ожидает ввода пользователя, обрабатывает этот ввод и повторяет процесс до тех пор, пока пользователь не выберет завершение (ввод 0).

int main() {

setlocale(LC\_ALL, "Russian");

int choice;

do {

//Выводим меню и считываем выбор пользователя

display\_menu();

scanf("%d", &choice);

handle\_menu\_choice(choice);

} while (choice != 0);

return 0;

}

Start\_time записывает начальное время для измерения времени сортировки. Внешний цикл проходит по всем элементам массива, кроме последнего. Внутренний цикл ищет минимальный (или максимальный, если order истинно) элемент в оставшейся части массива. Если минимальный (или максимальный) элемент не находится на текущей позиции, элементы меняются местами. end\_time записывает конечное время, и время сортировки вычисляется и выводится на экран.

//Данный блок кода сортирует массив методом выбора в прямом или обратном порядке в зависимости от выбора пользователя

clock\_t start\_time = clock();

for (int i = 0; i < count - 1; i++) {

int min\_index = i;

if (order) {

for (int j = i + 1; j < count; j++) {

if (numbers[j] > numbers[min\_index]) {

min\_index = j;

}

}

}

else {

for (int j = i + 1; j < count; j++) {

if (numbers[j] < numbers[min\_index]) {

min\_index = j;

}

}

}

if (min\_index != i) {

int temp = numbers[i];

numbers[i] = numbers[min\_index];

numbers[min\_index] = temp;

}

}

clock\_t end\_time = clock();

double time\_spent = (double)(end\_time - start\_time) / CLOCKS\_PER\_SEC;

system("cls");

//Выводим время за которое был отсортирован массив

printf("Время сортировки: %.3f секунд\n", time\_spent);

//Записываем отсортированный массив в файл

write\_file(output\_filename, numbers, count);

free(numbers);

system("PAUSE");

system("cls");

}

# **4.** **Схемы программы**

## **4.1 Блок-схема программы**

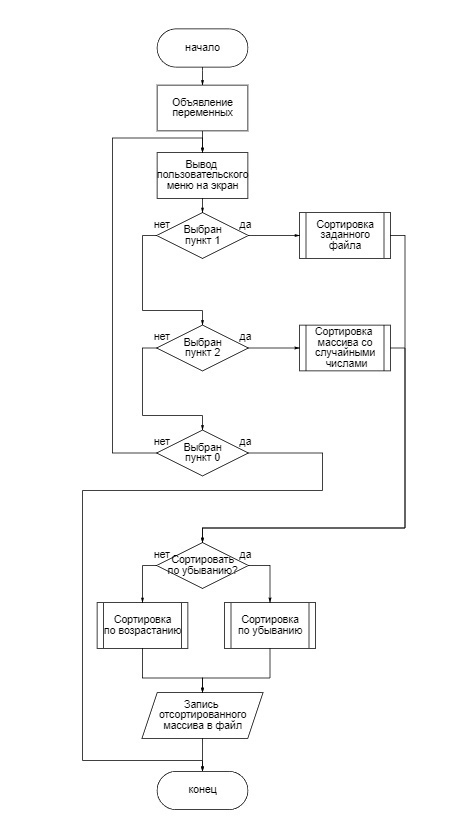


Рисунок 1 - блок-схема программы

## **4.2 Блок-схема алгоритма**



Рисунок 2 - блок-схема алгоритма

# **5. Тестирование программы**

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в Приложении А на рисунках А.1 - А.8.

Таблица 1 – Тестовый набор данных

|  |  |  |
| --- | --- | --- |
| № теста | Размер массива size | Время выполнения сортировки в секундах |
| 1 | 10 | 0.000 |
| 2 | 100 | 0.000 |
| 3 | 1000 | 0.001 |
| 4 | 10000 | 0.051 |
| 5 | 100000 | 5.071 |
| 6 | 200000 | 20.412 |
| 7 | 300000 | 45.777 |
| 8 | 400000 | 81.319 |

На основании анализа данных, полученных в результате тестирования алгоритма быстрой сортировки, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов увеличивается экспоненциально, то есть с увеличением количества элементов экспоненциально увеличивается время работы программы. Это можно увидеть на рисунке 3.

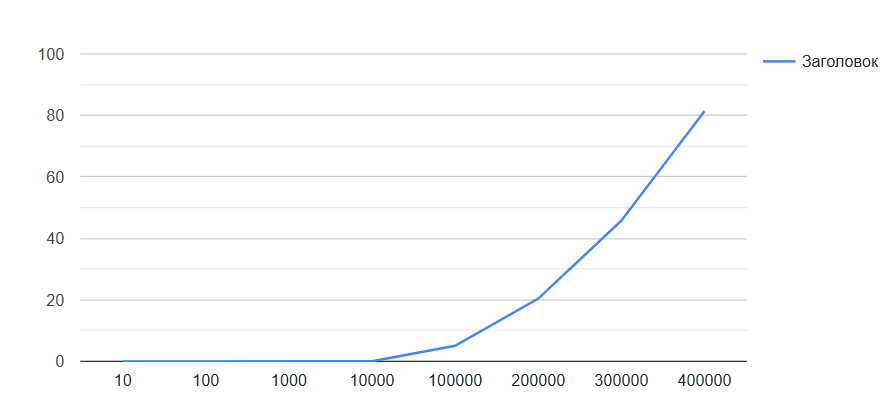


Рисунок 3 - результат тестирования

# **6. Отладка**

В качестве среды разработки была выбрана программа Microsoft Visual Studio, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ.

Для отладки программы использовались точки остановки и пошаговое выполнение кода программы, анализ содержимого локальных переменных.

Точки останова – это прерывание выполнения программы, при котором выполняется вызов отладчика. Отладчик является инструментом для поиска и устранения ошибок в программе, с помощью которого можно исследовать состояние программы.

Был использован метод бинарного поиска, он включает в себя разделение частей кода для упрощения процесса отладки. Это может быть особенно полезно, если причина ошибки находится в начале языка программирования, а фактическая ошибка ближе к концу.

Команда шаг с заходом (step into) выполняет следующую инструкцию в обычном пути выполнения программы, а затем приостанавливает выполнение программы, чтобы мы могли проверить состояние программы с помощью отладчика. Если выполняемый оператор содержит вызов функции, *шаг с заходом* заставляет программу перескакивать в начало вызываемой функции, где она приостанавливается.

# **7. Совместная разработка**

Во время работы над данной практикой наша бригада осуществляла совместную работу в GitHub. Каждый из бригады выполнял свои задачи: Куликовский В.А. делал работу с файлами, Усов А.С. проводил тестирования, Николаев А.В. реализовал алгоритм.

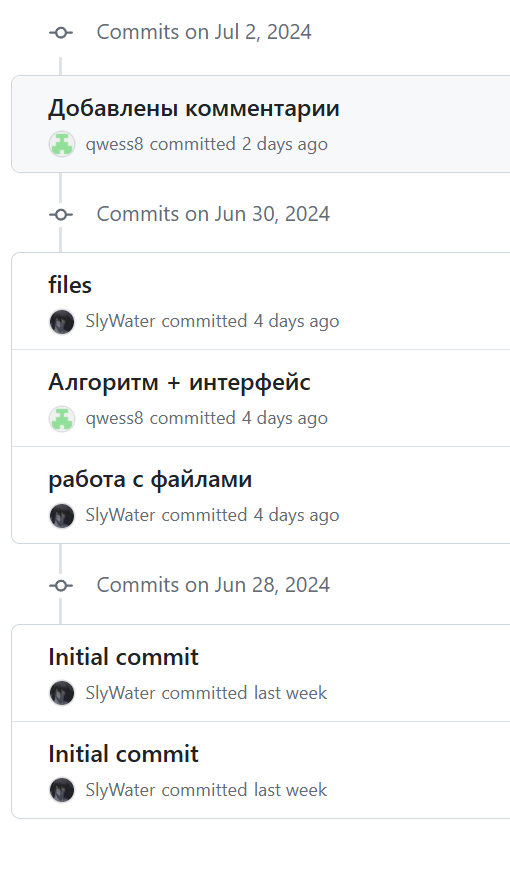


Рисунок 4 - совместная работа

Ссылка на удаленный репозиторий:

[SlyWater/Practice (github.com)](https://github.com/SlyWater/Practice)

# **Заключение**

При выполнении данной работы были получены навыки совместной работы с помощью сервисов GitHub, навыки использования программы Git Bash. Был изучен алгоритм сортировки вставками.

Я тщательно проанализировал лучшие и худшие случаи, которые встречались в ходе тестирования, чтобы получить всестороннее представление о возможных результатах. Для этого я собрал обширную выборку данных, включающую как успешные примеры, так и те, которые показали себя наименее эффективными. Такой всесторонний анализ позволил выявить ключевые факторы успеха и области, требующие улучшения.

При выполнении практической работы были улучшены базовые навыки программирования на языке С. Улучшены навыки отладки, тестирования программ и работы со сложными типами данных.

В дальнейшем программу можно улучшить путем подключения упрощающих реализацию данной сортировки библиотек и улучшения графического интерфейса.

# **Список используемой литературы**

1. ГОСТ 19.701 – 90 Схемы алгоритмов, программ, данных и систем.

2. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2- е издание.: Пер. с англ. – М.,2009.

3. Сортировка выбором [Электронный ресурс] – URL: [Сортировки выбором / Хабр (habr.com)](https://habr.com/ru/articles/422085/)

4. Сортировка выбором [Электронный ресурс] – URL: [Сортировка выбором — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BE%D0%BC)

# **Приложение А. Листинг программы**

Файл main.c:

#include "utils.h"

int main() {

setlocale(LC\_ALL, "Russian");

int choice;

do {

//Выводим меню и считываем выбор пользователя

display\_menu();

scanf("%d", &choice);

handle\_menu\_choice(choice);

} while (choice != 0);

return 0;

}

Файл utils.h:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <string.h>

#define REVERSE 1

#define NORMAL 0

void create\_file\_path(const char\* filename, char\* path);

// Функция для создания пути по шаблону files/%s.txt

void create\_path(char\* input\_path, char\* output\_path);

// Функция для создания путей к файлам ввода и вывода

int\* read\_numbers\_from\_file(const char\* filename, int\* count);

// Функция для считывания массива чисел из файла

void write\_file(const char\* output\_filename, int\* numbers, int count);

// Функция для записи массива в файл

void fill\_file\_with\_random\_numbers(const char\* filename, int count, int min, int max);

// Функция для заполнения файла случайными числами

void selection\_sort\_file(const char\* input\_filename, const char\* output\_filename, char order);

// Сортировка выбором

void sort\_by\_order(char\* input\_path, char\* output\_path);

// Функция для определения порядка сортировки

void display\_menu();

// Функция для вывода меню

void handle\_menu\_choice(int choice);

// Функция для обработки действия при выборе пункта меню

Файл utils.c:

#include "utils.h"

int\* read\_numbers\_from\_file(const char\* filename, int\* count) {

FILE\* file = fopen(filename, "r");

int capacity = 10;

int\* numbers = malloc(capacity \* sizeof(int));

int num\_count = 0;

char line[100];

//Данный блок кода построчно считывает числа из заданного файла и записывает их в массив

while (fgets(line, sizeof(line), file) != NULL) {

char\* cleaned\_line = strtok(line, "\n\r ");

if (cleaned\_line != NULL) {

int number;

if (sscanf(cleaned\_line, "%d", &number) == 1) {

if (num\_count >= capacity) {

capacity \*= 2;

int\* new\_numbers = realloc(numbers, capacity \* sizeof(int));

numbers = new\_numbers;

}

numbers[num\_count++] = number;

}

else {

fprintf(stderr, "Строка '%s' не является числом и будет пропущена.\n",

cleaned\_line);

}

}

}

fclose(file);

\*count = num\_count;

return numbers;

}

void fill\_file\_with\_random\_numbers(const char\* filename, int count, int min, int max) {

FILE\* file = fopen(filename, "w");

srand(time(NULL));

//Данный блок кода заполняет файл случайными числами

for (int i = 0; i < count; i++) {

int number = min + rand() % (max - min + 1);

fprintf(file, "%d\n", number);

}

fclose(file);

}

void write\_file(const char\* output\_filename, int\* numbers, int count) {

FILE\* output\_file = fopen(output\_filename, "w");

//Данный блок кода записывает отсортированный массив в файл

for (int i = 0; i < count; i++) {

fprintf(output\_file, "%d\n", numbers[i]);

}

fclose(output\_file);

}

void create\_file\_path(const char\* filename, char\* path) {

//Создаем путь к файлам ввода или вывода

snprintf(path, 50, "files/%s.txt", filename);

}

void create\_path(char\* input\_path, char\* output\_path) {

char input\_filename[40];

char output\_filename[40];

//Данный блок кода запрашивает у пользователя имена файлов для ввода или вывода

system("cls");

printf("Введите имя файла с несортированными числами (без расширения): ");

scanf("%255s", input\_filename);

create\_file\_path(input\_filename, input\_path);

printf("Введите имя файла с сортированными числами (без расширения): ");

scanf("%255s", output\_filename);

create\_file\_path(output\_filename, output\_path);

}

void selection\_sort\_file(const char\* input\_filename, const char\* output\_filename, char order) {

int count;

int\* numbers = read\_numbers\_from\_file(input\_filename, &count);

if (numbers == NULL) {

return;

}

//Данный блок кода сортирует массив методом выбора в прямом или обратном порядке в зависимости от выбора пользователя

clock\_t start\_time = clock();

for (int i = 0; i < count - 1; i++) {

int min\_index = i;

if (order) {

for (int j = i + 1; j < count; j++) {

if (numbers[j] > numbers[min\_index]) {

min\_index = j;

}

}

}

else {

for (int j = i + 1; j < count; j++) {

if (numbers[j] < numbers[min\_index]) {

min\_index = j;

}

}

}

if (min\_index != i) {

int temp = numbers[i];

numbers[i] = numbers[min\_index];

numbers[min\_index] = temp;

}

}

clock\_t end\_time = clock();

double time\_spent = (double)(end\_time - start\_time) / CLOCKS\_PER\_SEC;

system("cls");

//Выводим время за которое был отсортирован массив

printf("Время сортировки: %.3f секунд\n", time\_spent);

//Записываем отсортированный массив в файл

write\_file(output\_filename, numbers, count);

free(numbers);

system("PAUSE");

system("cls");

}

void display\_menu() {

//Отображаем меню действий, которые может выполнить пользователь

printf("1. Сортировка определенного файла\n");

printf("2. Сортировка файла с определенным количеством чисел\n");

printf("0. Выход\n");

printf("Выберите пункт меню: ");

}

void handle\_menu\_choice(int choice) {

int count, min, max;

char pathi[50], patho[50];

system("cls");

//Считываем, какое действие выбрал пользователь и выполняем его

switch (choice) {

case 1:

//Если пользователь выбрал сортировку массива числа к котрому лежат в заданном файле

create\_path(pathi, patho);

sort\_by\_order(pathi, patho);

break;

case 2:

//Если пользователь выбрал сортировку массива с случайными числами

printf("Введите количество чисел: ");

scanf("%d", &count);

printf("Введите минимальное значение: ");

scanf("%d", &min);

printf("Введите максимальное значение: ");

scanf("%d", &max);

create\_path(pathi, patho);

fill\_file\_with\_random\_numbers(pathi, count, min, max);

sort\_by\_order(pathi, patho);

break;

case 0:

//Выход из программы

printf("Выход из программы.\n");

break;

default:

printf("Неверный выбор. Пожалуйста, попробуйте снова.\n");

break;

}

}

void sort\_by\_order(const char\* input\_path, const char\* output\_path) {

int ch;

system("cls");

//Предагаем пользователю в каком порядке нужно отсортировать массив

printf("1. Обычный порядок\n2. Обратный порядок\nВыберите порядок сортировки: ");

scanf("%d", &ch);

switch (ch) {

case 1:

//Сортировка по возрастанию

selection\_sort\_file(input\_path, output\_path, NORMAL);

break;

case 2:

//Сортировка по убыванию

selection\_sort\_file(input\_path, output\_path, REVERSE);

break;

}

}